High-speed autonomous navigation of unknown environments using learned probabilities of collision

In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2014).

High-Speed Autonomous Navigation of Unknown Environments using Learned Probabilities of Collision

Charles Richter, John Ware and Nicholas Roy

***Abstract*— We present a motion planning algorithm for dynamic vehicles navigating through unknown environments. We focus on the scenario in which a fast-moving car attempts to navigate from a start location to a set of goal coordinates in minimum time with no prior information about the environment, building a map in real time from onboard sensor data. Whereas existing planners for exploration confine themselves to a conservative set of constraints to guarantee safety around unknown regions of the environment, we instead learn a hazard function from data, which maps the vehicle's dynamic state and current environment knowledge to a probability of collision. We perform receding horizon planning in which the objective function is evaluated in expectation over those learned probabilities of collision. Our algorithm demonstrates sensible emergent behaviors, like swinging wide around blind corners, slowing down near the map frontier, and accelerating in regions of high visibility. Our algorithm is capable of navigating from start to goal much more quickly than the conservative baseline planner without sacrificing safety. We demonstrate our algorithm on a 1:8-scale high-performance RC car equipped with a planar laser range-finder and inertial measurement unit, reaching speeds of 4m/s in unknown, indoor spaces. A video of experimental results is available at:** `http://groups.csail.mit.edu/rrg/nav_learned_prob_collision`.

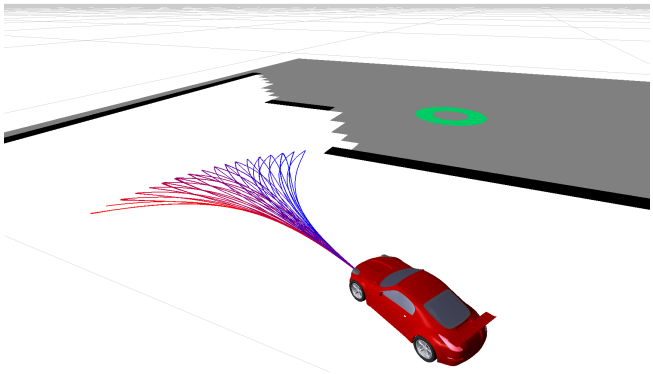


Fig. 1: A fast car navigating a partially observed environment toward a goal location (green circle). Observed walls are shown in black and unobserved regions are gray. This car could employ a risky strategy of driving fast toward the goal into the unknown region, or travel slowly in order to observe the unknown regions safely. We present a strategy learned from data, which outperforms both of these options.

I. INTRODUCTION

In recent years, robots and autonomous vehicles have enjoyed considerable success maneuvering quickly in environments where a complete prior map is available. However, many important tasks that would be ideal for robotic applications occur in areas where prior information is unavailable. These cases include search and rescue missions in disaster zones, military operations in outdoor wilderness environments, or personal service applications in cluttered spaces where the layout of objects changes frequently. In these cases, it is infeasible to obtain a complete prior map in order to navigate quickly, and the agent must instead make decisions using its onboard sensing capabilities in real time.

While many robots navigate without prior knowledge of their environment, they typically constrain themselves to plan motions within the known portion of the environment, and move slowly enough to stop before entering unknown space. However, for vehicles attempting to navigate quickly, these constraints may be overly conservative. Experience can be used to make reasonable predictions about outcomes of actions that traverse some unmapped space based on the dynamics of the vehicle and learned environmental structure.

In this paper, we focus on the navigation problem for a high-speed dynamically constrained vehicle moving from an initial state to a specified set of global coordinates in minimum time. Our solution strategy is to *learn* which scenarios should be avoided by mapping features of vehicle state and environment knowledge to probabilities of collision. At runtime, these probabilities are used to select actions that minimize cost in expectation over the experience of the vehicle. Having learned a function representing probability of collision, our planner exhibits driving techniques such as swinging wide around blind corners, slowing down near the map frontier, and accelerating in regions of high visibility.

The problem of learning to drive in unknown environments has been studied in a wide range of previous work [10], [22], [20]. However, most previous work has focused on learning to drive in the parts of the environment that have already been observed. The contribution of this paper is to show how a model of risks associated with a partially unknown map can be learned from data, and incorporated into a motion planner to select trajectories that allow the robot to move safely towards the goal faster than existing strategies. Effectively, this model allows the vehicle to anticipate underlying structure in unobserved portions of environment, overcoming the traditional limitations of incomplete map knowledge to enable high-speed navigation.

A. Motivating Example

Figure 1 illustrates a scenario in which a vehicle is attempting to reach a goal location (green circle) that lies around a corner in an unobserved region of the environment. From the vehicle's point of view, the observed walls (drawn in black) occlude unknown space (drawn in gray), leaving

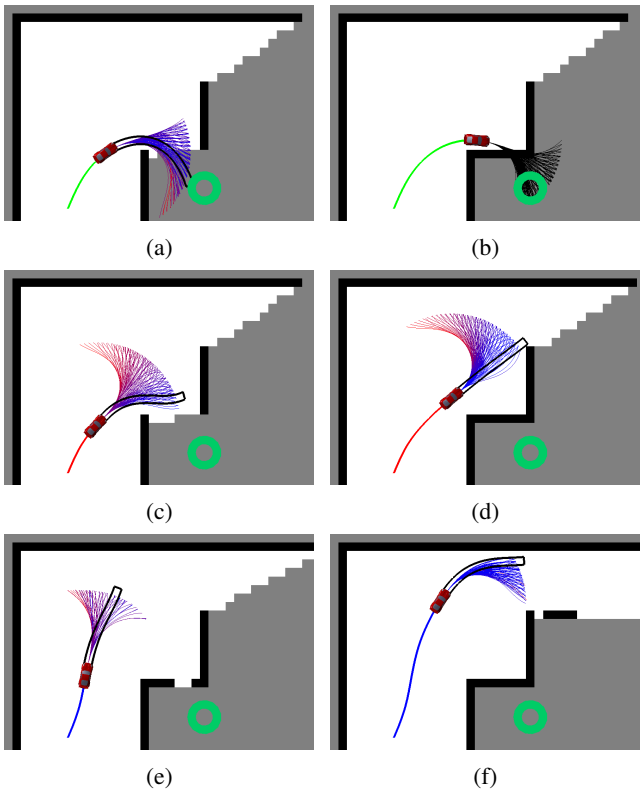


Fig. 2: Execution of three planning strategies from the same initial state depicted in Figure 1: Greedy (a,b), conservative (c,d), and learned (e,f). Images (a, c, e) show their respective planners after 0.4 seconds of navigation from the initial state, while images (b, d, f) show the three planners after 1.0 second. Colored lines represent the set of possible trajectories. The chosen actions are outlined in black. Note that after 0.4 seconds, the maps of the greedy and conservative planners are identical, but path chosen by the learned planner enables the senor to uncover more map information.

a potentially traversable opening in the map. The vehicle cannot determine the occupancy of those regions before rounding the corner. Figure 2 shows three possible navigation strategies for the car starting from this initial state.

The greedy strategy (2a, 2b) assumes that unobserved regions are unoccupied and heads directly toward the goal at full speed. However, upon rounding the blind corner, the vehicle is able to take measurements of the unknown region and discovers a wall blocking its path. Due to constraints on its braking and steering, the car is now unable to avoid collision with the wall. The vehicle's possible trajectories in this case are colored black since they all intersect the wall and are therefore infeasible.

The conservative strategy (2c, 2d) assumes unknown regions are occupied until they are measured to be free, and limits velocity to maintain the ability to stop before the nearest obstacle or unknown region. Following this strategy, the car drives directly toward the goal at a slow pace, and is able to turn away once it observes walls around the corner.

Our planner, with a learned model of expected costs (2e, 2f), selects actions that probabilistically balance the risks of driving near obstacles and unknown regions of the map with the benefits of moving quickly toward the goal. In order to perform this trade-off, we need to be able to evaluate expected costs of motions through partially unknown maps of the environment. To do so, we estimate collision probabilities in a manner that is distinct from prior work in that we model the risks associated with traversing unknown regions. In this paper, we show that using this new type of learned probability model allows the vehicle to make much more progress than a conservative strategy in the same amount of time, without the risks associated with a greedy strategy.

II. PROBABILISTIC RECEDING HORIZON PLANNING

Motion planning in partially observed environments requires that uncertainty in action execution be considered, since the cost associated with a trajectory depends on the unknown portion of the environment. Specifically, when planning a high-speed trajectory into an unknown region of the environment, the success or failure of that trajectory, and hence its true cost, depends on the occupancy of the unobserved region. Therefore, it is impossible to evaluate the true cost of a trajectory at runtime.

Rather than minimizing the true cost of an action $J(a)$, we instead choose actions that minimize cost in expectation:

$$a^* = \underset{a \in A}{\operatorname{argmin}} \; E_{map}[J(a)] \tag{1}$$

where a is a trajectory from the current location of the vehicle all the way to the goal, and $E_{map}[J(a)]$[1] is the expected cost of the action with respect to the map. It is unlikely that we could learn to predict the probability of collision all the way to the goal, but there is some horizon over which we may be able to learn an accurate predictor. We use a receding horizon formulation, where we compute the expected cost up to a finite planning horizon, and then use a heuristic to estimate the path cost from that point to the goal:

$$a^* = \underset{a \in A}{\operatorname{argmin}} \; E_{map}[J(a)] + h(x_a) \tag{2}$$

This function consists of two components: the expected cost of the action to the planning horizon, and the heuristic estimate of cost-to-go, $h(x_a)$, from the state at the end of the action. The set A includes a subset of all possible trajectories from the current state to the planning horizon, and is intended to span the vehicle's maneuvering capabilities.

Although we are operating in an unknown environment, we assume a SLAM process is inferring a map from sensor data. Most SLAM processes do not provide a single map estimate but provide a probability distribution over the map, with greater uncertainty in the parts of the map with less sensor data. Because the map is not known exactly, we cannot compute the cost of a trajectory $J(a)$ exactly but need to compute the expected cost with respect to the map distribution $p(map)$. There are several ways to model $E_{map}[J(a)]$,

[1]To avoid confusion: The map distribution is over maps, not the *maximum a posteriori* distribution that is often referred to as p_{MAP} in machine learning.

depending on our assumptions. We could make the naïve assumption that unobserved regions of the map are free and traversable, or the conservative assumption that unobserved regions are occupied. Both of these assumptions are likely to be incorrect. Unfortunately, computing the expected cost exactly is equally difficult, because the map distributions are typically very high dimensional, and hence difficult to integrate, and also usually contain independence assumptions that allow efficient inference of the lower moments of the distribution but do not capture the entire distribution accurately enough to use for expected cost calculations.

An alternative approach is possible, since computing the expectation with respect to the full map distribution is actually unnecessary. In our planning problem, costs are due to either successfully completing the trajectory or experiencing a collision. The expectation can then be computed over the total probability of collision along the trajectory, $p_{total}(a)$:

$$E_{map}[J(a)] = p_{total}(a) \cdot J_{collision} + (1 - p_{total}(a)) \cdot J_{free}(a) \quad (3)$$

The expected value of a trajectory in equation (2) is a weighted sum of the free-space cost of a trajectory $J_{free}(a)$ and the penalty we assign to a collision $J_{collision}$. The weighting applied to these two terms is a function of the total probability that the vehicle will collide along the trajectory.

For the minimum-time problem, J_{free} is simply the time duration of the action. To compute p_{total} for any given trajectory, we use the quantity $p_{collision}(x_t, m_t)$, which represents the probability that the vehicle will enter an inevitable collision state (ICS) between the current time t and the subsequent planning iteration at $t + \Delta t$. This quantity is a function of the vehicle state, x_t, and the state of the vehicle's map of the environment, m_t, at that time. We consider the vehicle to be in an ICS if, on its next planning iteration, all of the possible trajectories the vehicle could execute collide with known walls or obstacles in the environment.

If we know $p_{collision}(x_t, m_t)$ at every discrete time increment t along a trajectory, then we can estimate the probability that the vehicle will enter an ICS along that trajectory:

$$p_{total} = 1 - \prod_{t=0}^{T} 1 - p_{collision}(x_t, m_t) \quad (4)$$

where T is the trajectory duration. Intuitively, the probability of collision approaches one as its duration increases for any given value of $p_{collision} > 0$. Therefore, it is important to compare trajectories of equal duration or otherwise correct for differing lengths. Equation (4) is an instance of the geometric distribution, used to estimate the cumulative effect of discrete risks represented by the discrete hazard function $p_{collision}$. This type of model representing risk over the lifetime of an agent is common in reliability engineering.

III. LEARNING PROBABILITIES OF COLLISION

Even with the expectation computed over the simpler binary probability distribution, it is still typically not possible to evaluate $p_{collision}$ from a map. The probabilities of environmental structure inside the unknown regions are typically very poor predictors of collision probability. Nevertheless, there are features of the environment and vehicle state that suggest how safe or how dangerous any given trajectory is likely to be. For instance, the vehicle is likely to be safe traveling down a straight, empty hallway with high visibility in its direction of travel. However, when it nears the turn at the end of the hallway, it will be unable to take a measurement of the environment ahead until it rounds the corner. In this scenario, the vehicle may risk a high probability of collision if it rounds the corner too quickly.

We estimate these probabilities of collision using a learned function, which maps features $\phi(x_t, m_t)$ of the state and environment map to a scalar probability of collision:

$$f_{collision}(\phi(x_t, m_t)) \approx p_{collision}(x_t, m_t) \quad (5)$$

We learn the function $f_{collision}$ from simulation data by repeatedly simulating the car starting from random points in space and allowing it to navigate toward the goal by greedily selecting the action that descends the heuristic function most rapidly. We record the features encountered by the vehicle at every time step, and associate an outcome (`collision` or `non-collision`) with each feature record. If the vehicle succeeds in reaching the goal, then the feature history associated with that trial is labeled `non-collision`. If a collision occurs during simulation, then the feature values preceding the collision are labeled as `collision`.

We use multivariate logistic regression to fit a sigmoid (logistic) function to approximate the binomial distribution of outcomes throughout the feature space. The probability of collision for a point in feature space is given by the logistic function of features ϕ_i and associated weights w_i:

$$f_{collision}(\phi) = \frac{1}{1 + \exp(-w_0 - \sum_{i=1}^{N} w_i \phi_i)} \quad (6)$$

Logistic regression is used to find weights w_i such that $f_{collision}$ most closely fits the observed ratio of `collision` to `non-collision` throughout feature space.

Five features were used to predict collision probabilities, denoted $\phi_1 \ldots \phi_5$. Let $d_{occ.}$ and $d_{unk.}$ denote the distances to the nearest obstacle and frontier, and let $\mathbf{r}_{\mathbf{occ.}}$ and $\mathbf{r}_{\mathbf{unk.}}$ denote the vectors between the vehicle position and the nearest obstacle and frontier, respectively. Let $\mathbf{v}$ denote the vehicle's velocity vector. The features are:

- Distance to obstacle: $\phi_1 = d_{occ.}$
- Distance to frontier: $\phi_2 = d_{unk.}$
- Velocity toward obstacle: $\phi_3 = \mathbf{v} \cdot \mathbf{r}_{\mathbf{occ.}} / \|\mathbf{r}_{\mathbf{occ.}}\|$
- Velocity toward frontier: $\phi_4 = \mathbf{v} \cdot \mathbf{r}_{\mathbf{unk.}} / \|\mathbf{r}_{\mathbf{unk.}}\|$
- Scalar vehicle speed: $\phi_5 = \|\mathbf{v}\|$

These features were selected to enable basic reasoning about the vehicle's motion with respect to the geometry of the known environment structure as well as the unknown regions. While many more features could be added to this list, feature selection lies outside the scope of this work.

IV. HEURISTIC COST-TO-GO FUNCTION

The second term in equation (2), $h(x_a)$, is a heuristic function returning the estimated cost-to-go from the state x_a at the end of each possible trajectory to the goal. The

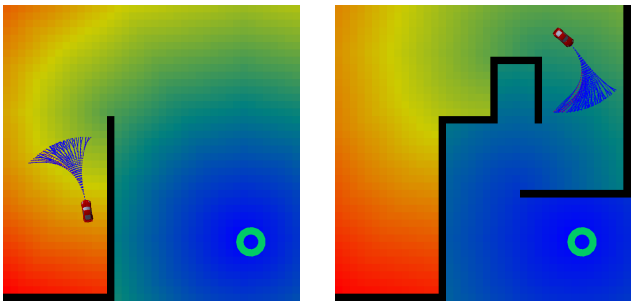


Fig. 3: Numerical potential field heuristic computed over a 2D graph representing x, y-locations in the map. Cost-to-go distance increases from blue to red.

heuristic function is a numerical potential field [1] computed over a two-dimensional graph representing x, y-positions, and assumes that the unobserved space is free. Thus, the heuristic function provides high-level guidance, respecting the walls and obstacles that are currently known, which would not be captured by using Euclidean distance to the goal as a heuristic. Figure 3 illustrates the heuristic function at two different stages in the navigation of an unknown space, showing the changes in estimated cost-to-go as the map structure is revealed.

V. RESULTS

We first describe simulation results, with all training and testing being performed on a ground robot model with a laser range-finder and input-constrained kinematic car dynamics (discussed in Section VI). The data used to learn $f_{collision}$ were generated from 250 simulation runs (5125 data points in feature space) on the map shown in Figure 4. This training set was generated in approximately 15 minutes, while logistic regression was completed in 0.03 seconds, highlighting the computational efficiency of our procedure. The learned collision probability model was then provided to the planner for testing simulations to measure time-to-goal and success rate after learning. Finally, we discuss experimental results for the actual RC car carrying a laser range-finder, using the model of collision probability that was learned and tested in our simulation results.

A. Learned Probabilities of Collision

Figures 5 shows one characteristic slice of the 5-dimensional function $f_{collision}$, displaying ϕ_2 (distance to nearest frontier) and ϕ_5 (speed), with the other features held constant. The shape of this function indicates that traveling at high speed near the frontier carries a high probability of collision, but either slowing down or moving away from the frontier can substantially reduce that risk. These features capture the danger in the example depicted in Figure 1, and illustrate why the learned planner in that example kept a safe distance from the frontier while maintaining a high speed.

The actual data points used to perform regression are illustrated as a histogram in Figure 5. Red circles indicate `collision` points and blue circles indicate `non-collision` points. The area of each circle is proportional to the number of data points landing in that region of feature space. Since this plot illustrates only two of the five dimensions of the logistic function, the data points themselves need not actually match the shape of the sigmoid shown here. However, the data points serve to illustrate the region of feature space where collisions are most likely, and the results agree with the intuitive result that driving at high speed near unknown regions carries a greater risk than driving more slowly or at a safer distance from the frontier.

B. Learned Planner Performance

Figure 4a shows the execution results of a planning problem without the use of learned weights. This case corresponds to a greedy policy of simply choosing the action that descends the global cost-to-go function as rapidly as possible. In this case, only several simulations succeeded in finding the goal. Collision states are illustrated with red dots, and the anticipated trajectory beyond each collision state is shown in gray. These gray trajectories appeared to be feasible until an unknown portion of the environment was observed to be occupied, rendering the trajectory infeasible. Collision states often occur near corners where the vehicle attempts to round the corner as tightly as possible, but is going too fast to steer away to avoid collision once it observes the structure of the environment around the corner.

A second, conservative planner was implemented for comparison. This conservative planner drives in the direction that most directly descends the heuristic function, but it is constrained to plan paths entirely within the known free portion of the environment. Moreover, it is constrained to limit its velocity such that its stopping distance is less than both the distance to the nearest occupied cell and the distance to the nearest unknown cell in the environment. These constraints guarantee that the planner can come to a stop if necessary. Figure 4b shows the execution of the conservative planner. This planner approximately follows the shortest route from start to goal that is kinematically feasible.

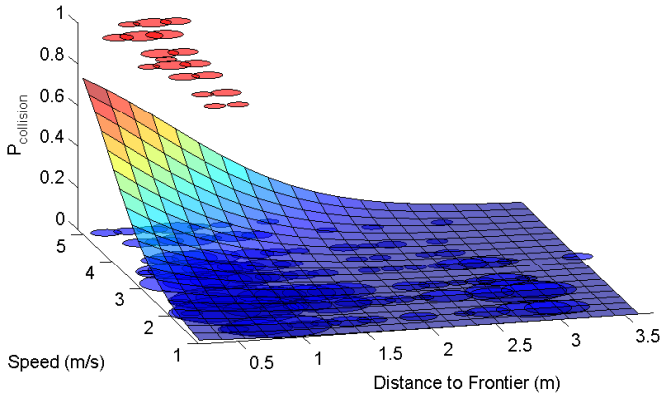
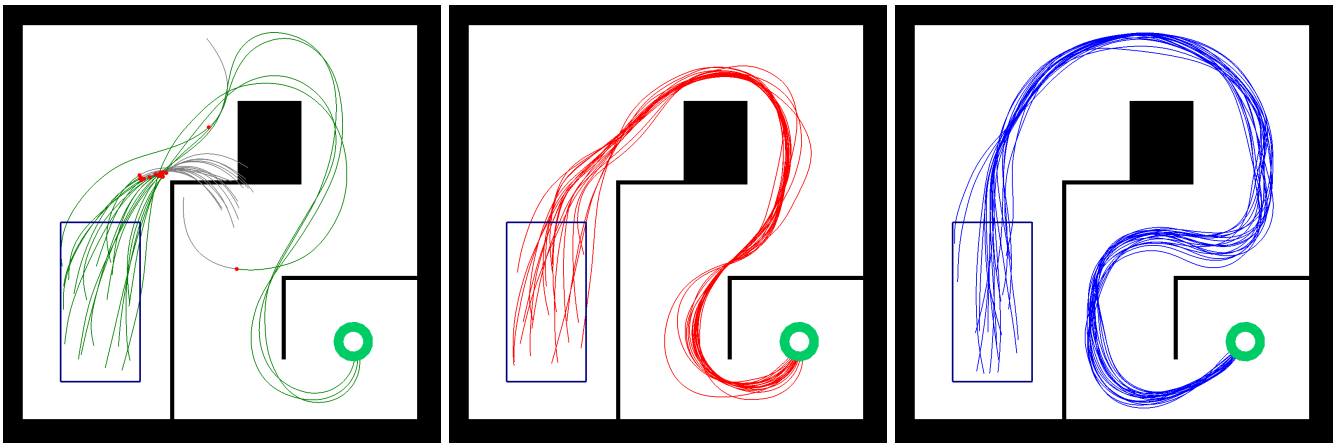


Fig. 5: Regression results for the pair of features ϕ_2 (distance to nearest frontier) and ϕ_5 (speed).



(a) Greedy planner. (b) Conservative planner. (c) Planner after learning.

Fig. 4: Comparison between 20 simulations of the greedy (a), conservative (b), and learned (c) planning algorithms, initialized randomly within the blue box. The goal is indicated by the green circle. The greedy planner descends the heuristic function as rapidly as possible, often entering ICSs with yet-unseen obstacles. ICSs are indicated with red dots. Intended future trajectories are drawn in gray. The conservative planner travels a shorter distance around the environment than the learned planning algorithm, but must maintain a lower speed to guarantee safety with respect to observed and unobserved obstacles.

	Greedy Planner			Conservative Planner			**Learned Planner**		
Environment	Time (s)	Distance (m)	Success	Time (s)	Distance (m)	Success[2]	Time (s)	Distance (m)	Success
Training Map	7.1 ± 0.8	24.2 ± 3.6	20%	14.9 ± 0.7	20.7 ± 1.1	100%	9.1 ± 0.4	26.0 ± 1.3	100%
Testing Map 1	6.5 ± 0.7	27.7 ± 1.2	32%	18.6 ± 0.7	27.6 ± 1.5	96%	13.7 ± 0.8	33.1 ± 1.5	94%
Testing Map 2	3.3 ± 0.5	12.8 ± 2.0	46%	8.4 ± 0.3	12.3 ± 0.5	100%	6.8 ± 0.4	13.9 ± 0.4	100%
Testing Map 3	6.1 ± 3.0	24.0 ± 12.9	12%	10.0 ± 0.6	13.3 ± 1.0	92%	7.9 ± 0.5	17.6 ± 1.1	100%

TABLE I: Results for greedy, conservative and learned planners on the training map and three test maps that were not used for training. Data represent 50 trials per experiment. In each scenario, the learned planner reached the goal considerably faster than conservative planner without compromising safety. Times and distances are reported with their standard deviations.

Finally, our planning strategy using learned probabilities of collision was simulated in the training environment as well as a set of testing environments. Simulation results from the learned planner are shown in Figure 4c. The major qualitative difference in behavior is that the learned planner follows a much longer path from start to goal, whereas the greedy and conservative planners both move along the shortest route. The other major difference is in the vehicle speeds chosen by the planner. Generally, the greedy strategy applies full throttle and steers along the heuristic gradient direction. Conversely, the conservative planner usually travels slowly due to satisfy its stopping distance constraint. The learned planner, however, modulates its speed significantly, slowing down near frontiers and obstacles.

Results were collected from 50 trials of each planner in the training environment as well as three different maps that were not used for training. The greedy planner succeeded rarely, but reached the goal in very little time when it did. The conservative planner traveled shorter paths from start to goal than the learned planner, but did so at a lower speed. Finally, the learned planner had the greatest success percentage overall among the three planners. Furthermore, it navigated considerably faster than the conservative planner despite its longer distance traveled.

The qualitative differences between Figures 4b and 4c are significant. While the conservative planner drove to the inside of every curve, the learned planner exhibited a strong preference for swinging wide around blind corners in order to distance itself from the map frontier. This behavior effectively allows the vehicle to project its sensor horizon much farther forward, allowing it to plan higher speed paths with confidence due to the increased map knowledge. A second benefit of this behavior is that by swinging wide, the vehicle allows itself maximal room to make a turn, which may be necessary depending on the structure of the environment beyond the frontier.

As shown in Table I, our learned planner was equally successful in a series of testing environments that differed from the training environment as it was in the training environment. These results suggest that our features and learned probability function are not specific to the training environment and capture essential driving strategies that are useful across many environments.

[2]We constrain all planners to move forward on every planning step. Therefore, while the conservative planner *could* safely come to a stop to avoid collision, it may still enter collision states in some confined environments due to kinematic constraints given that it must keep moving.

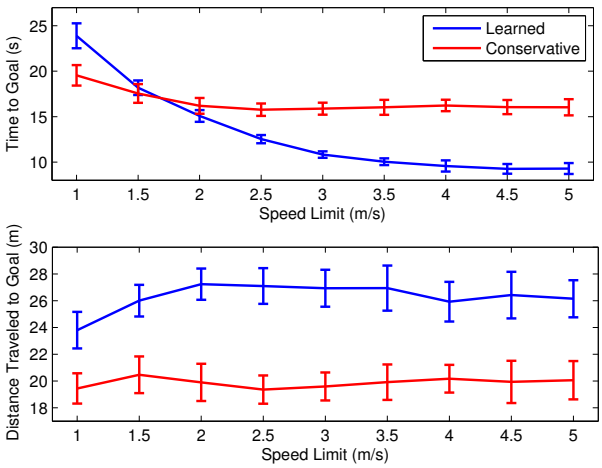


Fig. 6: Time (top) and distance (bottom) required to navigate from start to goal as a function of maximum allowed velocity in the environment pictured in Figure 4. Error bars indicate ± 1 standard deviation. The results of the conservative planner are shown in red and the learned planner in blue. The differences in distance traveled can be observed visually in Figures 4b, 4c.

C. Performance as a Function of Speed Limit

To highlight the differences between the learned and conservative planners, we ran simulation trials using both planners in which we increased the maximum allowable speed from $1m/s$ to $5m/s$. The results are illustrated in Figure 6. When limited to low speed, both planners travel approximately at the speed limit, so the conservative planner's preference for traveling the shortest path within the known map allows it to reach the goal sooner. However, as the speed limit is increased, the conservative planner becomes limited by the stopping distance constraint rather than the speed limit while the learned planner continues to utilize the additional speed. Above $4m/s$ the improvement plateaus, indicating that the learned planner has reached the maximum average speed at which it is confident navigating safely in this particular environment even though it is allowed to travel faster.

D. Learning Weights with Different Dynamics Models

The dynamics of the vehicle have a strong effect on its ability to evasively maneuver to avoid collision. In addition to the minimum turning radius, the curvature rate, acceleration, and braking constraints strongly impact the available actions from a given state. Therefore, while our results generalize across environments, it is important to perform learning on the vehicle model that will be used during execution. Figure 8 shows the set of available actions from a particular state for four different dynamics models that differ only in their acceleration and turning constraints. This figure shows that a more tightly constrained vehicle has a narrower range of actions available to avoid collision, whereas a less constrained vehicle can slow down or swerve easily.

We performed learning trials using dynamics models subject to these four sets of constraints. Figure 7 shows that optimal weights in the probability function $f_{collision}$ converge to significantly different values for different dynamics models, indicating that the modes and frequencies of collision differ between the dynamics models and that the learning process identifies these non-trivial differences.

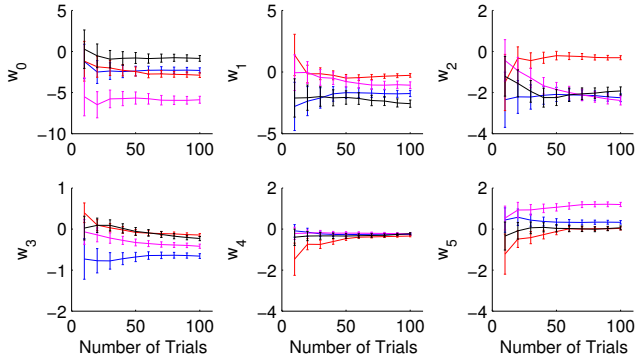


Fig. 7: Convergence of weights $w_0, \ldots, w_5$ in $f_{collision}$ for the different dynamics models depicted in Figure 8: less constrained (black), nominal (blue), more constrained (magenta) and most constrained (red). Error bars indicate standard error estimates returned by the regression algorithm.

VI. ACTUATOR-CONSTRAINED CAR MODEL

For the simulation and experimental results given in this paper, we employed a predictive motion model of a car with constraints on curvature, curvature rate, acceleration and braking. This model assumes zero slip between the wheels and the driving surface, and is intended to capture the types of actions that the car may take. States in the model are x, y, ψ, k, v, denoting the x,y-position, heading, curvature and velocity of the vehicle, respectively. The dynamics evolve according to:

$$\dot{x} = v \cdot \cos(\psi) \tag{7}$$

$$\dot{y} = v \cdot \sin(\psi) \tag{8}$$

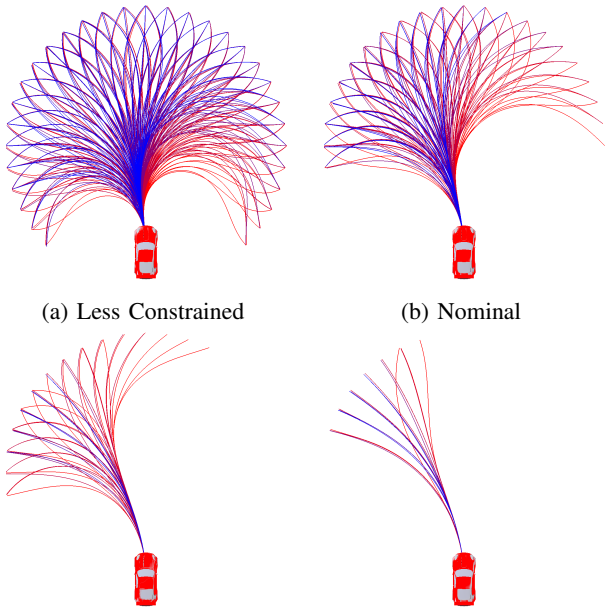
$$\dot{\psi} = k \cdot v \tag{9}$$

$$\dot{k} = (k_{cmd} - k)/dt, \quad s.t. \quad |k| \leq k_{max}, \quad |\dot{k}| \leq \dot{k}_{max} \tag{10}$$

$$\dot{v} = (v_{cmd} - v)/dt, \quad s.t. \quad b_{max} \leq \dot{v} \leq a_{max} \tag{11}$$

where the inputs to the system are the desired curvature (k_{cmd}) and desired velocity (v_{cmd}). This model differs from a Dubins car in that inputs are limited. While we are free to command a curvature value arbitrarily, which corresponds to an angle of the two front steering wheels, the curvature rate constraint ($\dot{k}_{max}$) will limit the rate of change of steering angle so that time must elapse before the wheels reach their desired angle. We set the maximum curvature rate to enforce a lock-to-lock steering time of 2 seconds. Limits are also placed on the acceleration (a_{max}) and braking (b_{max}).

In order to compute the action sets illustrated in Figure 8, a uniform sampling of terminal states was selected in an arc around the vehicle, with a variety of headings and curvatures to produce a rich set of curved trajectories. Then, a Newton method root finding routine was performed to solve for the



(a) Less Constrained (b) Nominal

(c) More Constrained (d) Most Constrained

Fig. 8: Trajectories from an initial state with $3m/s$ velocity and $0.5m^{-1}$ curvature (left-hand turn) using four dynamics models. Models used in (a)-(d) differ in k_{max}, $\dot{k}_{max}$, a_{max} and b_{max}. To enhance visibility, these images show only a portion of the hundreds of trajectories precomputed for any given state.

sequence of vehicle control inputs that would take the vehicle from its initial state to each of the terminal states. This procedure is outlined in [9] and represents the notion of a body-centered graph of nodes that are reachable from a given initial state. Using the body-centered approach allows the pre-computed set of actions to be transposed to any position and heading in the map, eliminating the burden of optimizing trajectories in each planning step.

VII. RC CAR EXPERIMENTS

We performed experiments on an autonomous 1:8 scale radio-controlled car equipped with an onboard computer with a dual-core Intel Core i7 processor, a Microstrain inertial measurement unit and Hokuyo planar laser range-finder. State estimation was performed onboard the vehicle using an IMU-driven extended Kalman filter with the filter formulation presented in [3], using a scan-matching algorithm on the laser data to provide relative odometry measurements. The laser data were also used to populate a grid cell map of the environment in real time, and this map was used for planning. Figure 9 shows the car used in the experiments, with the laser, IMU and computer onboard.

Figure 10 shows the partial map produced by the car on an experimental run through an indoor space, along with the set of available actions from the states depicted. In this environment, the car reached a velocities up to $4m/s$. A video of experimental results is available at: `http://groups.csail.mit.edu/rrg/nav_learned_prob_collision`.

Fig. 9: 1:8 scale RC car used in experiments, with Hokuyo laser range-finder, Microstrain IMU and computer.

VIII. RELATED WORK

Autonomous driving capabilities for full-sized vehicles have reached the point of safe driving in off-road and urban driving scenarios [23], [15], [17], [22], [20], using a variety of planning techniques including graph search, model-predictive control, and sampling-based methods [14], [6]. However, unlike our scenario, these examples rely heavily on prior knowledge of the map and leverage the road structure for perception and planning. Full-sized autonomous vehicles have also navigated unstructured environments such as parking lots, where the structure of roadways does not apply [12], [5], [4], though driving in these scenarios was not designed to be dynamic or to reason about unobserved regions of the environment, as was our objective. Similarly, planetary rovers now perform some mapping using onboard sensors and computation, though their trajectories are typically restricted to the mapped parts of the environment [7], [24]. In [2], a sampling-based planner was developed for kinodynamic systems in unknown environments, which guaranteed safety without reasoning about the unknown regions of the environment, similar to our conservative planner.

Learning has been applied to autonomous driving in a variety of cases that focus on perception, including Learning Applied to Ground Robots (LAGR) [10], [8], [13], which focused on extending the perception horizon for planning rather than planning with respect to occluded space, as in our work. The ALVINN driving system demonstrated steering using a neural network [19], while supervised learning has been used to select driving speed in desert terrain to trade shock on the system vs. speed [21], and non-parametric learning has been applied to aid path planning on sloped off-road terrain [11]. However, none of these cases explicitly reason about the outcomes of actions with respect to unobserved regions of the map. In the domain of RC cars, neural networks and monocular depth estimation coupled with policy search have been used for high-speed collision avoidance [16], [18], but neither system formed a comprehensive solution for navigation and control like the one we describe in this paper.

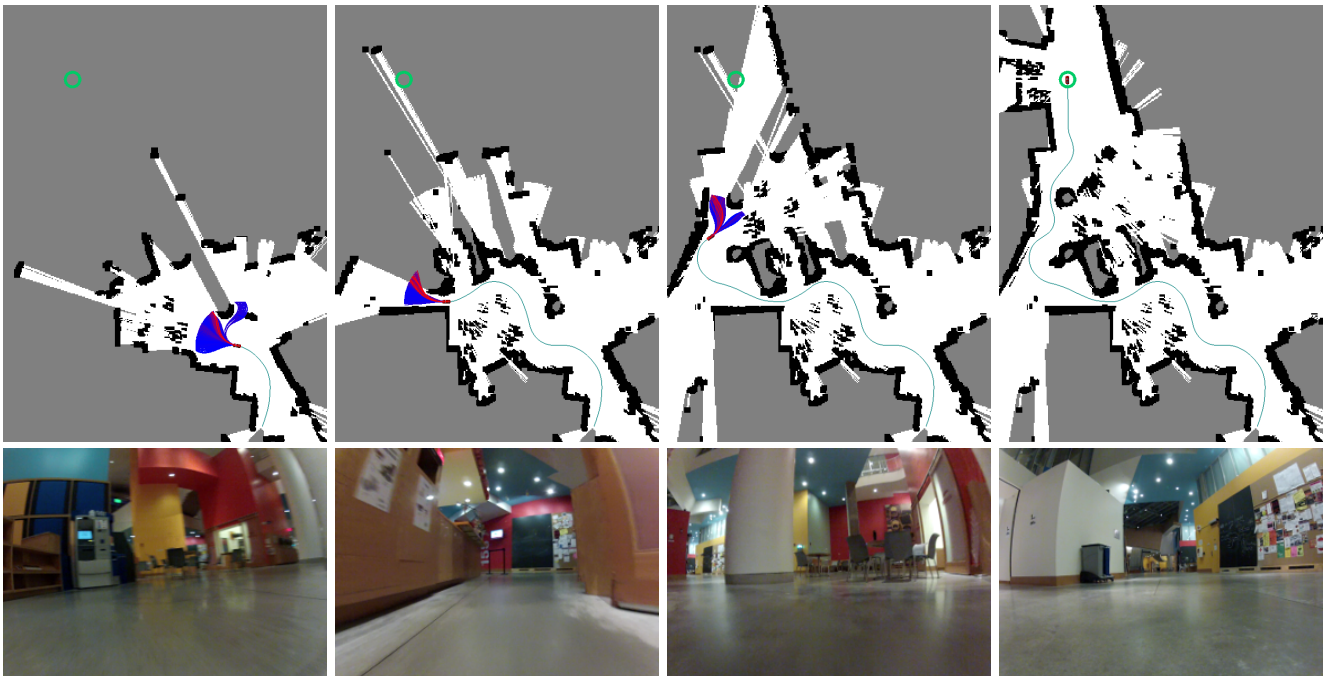


Fig. 10: Experimental trial driving up to approximately $4m/s$ through a lab space in the Stata Center (MIT) covering $60m$ of unknown, unstructured space. The goal (green circle) was not visible until nearly the end of the experimental run.

IX. CONCLUSIONS AND FUTURE WORK

We have demonstrated a motion planning algorithm for high-speed vehicles navigating in unknown environments. Our algorithm minimizes the expected cost of each trajectory over the experience of the vehicle, leading to high speed motions in situations where it is safe to move at high speed, while slowing down and selecting longer paths in situations where it is safer to do so. In this sense, our algorithm performs a principled probabilistic trade-off between risk and reward. The behaviors that emerge from our algorithm and learning process are very natural, intelligent behaviors that are characteristic of experienced drivers, and these behaviors arise naturally from relatively little of data.

Since our approach is agnostic to the the vehicle model and sensing modality, future work will involve extensions to quadrotors and fixed wing aircraft, which may necessitate lighter sensors such as cameras. It is unclear whether the same features will be effective across platforms, and the general problem of selecting the most effective set of features remains an unsolved challenge. Learning features themselves is one area of future work. Another extension will be to investigate the types of training strategies that result in the most accurate models for collision probability.

REFERENCES

[1] J. Barraquand, et. al. Numerical potential field techniques for robot path planning. *Systems, Man and Cybernetics, IEEE Transactions on*, 22(2):224–241, 1992.

[2] K. E. Bekris and L. E. Kavraki Greedy but safe replanning under kinodynamic constraints. In *Proc. ICRA*, 2007.

[3] A. Bry, et. al. State estimation for aggressive flight in GPS-denied environments using onboard sensing. In *Proc. ICRA*, 2012.

[4] D. Dolgov and S. Thrun. Autonomous driving in semi-structured environments: Mapping and planning. In *Proc. ICRA*, 2009.

[5] D. Dolgov, et. al. Path planning for autonomous vehicles in unknown semi-structured environments. *IJRR*, 29(5):485–501, 2010.

[6] D. Ferguson, et. al. Motion planning in urban environments: Part I. In *Proc. IROS*, 2008.

[7] S.B. Goldberg, et. al. Stereo vision and rover navigation software for planetary exploration. In *Aerospace Conf. Proceedings, IEEE*, 2002.

[8] A. Howard. Real-time stereo visual odometry for autonomous ground vehicles. In *Proc. IROS*, 2008.

[9] T. Howard and A. Kelly. Optimal rough terrain trajectory generation for wheeled mobile robots. *IJRR*, 26(2):141–166, 2007.

[10] L. D. Jackel, et al. The DARPA LAGR program: Goals, challenges, methodology, and phase I results. *JFR*, 23(11-12):945–973, 2006.

[11] S. Karumanchi, et al. Non-parametric learning to aid path planning over slopes. *IJRR*, 29(8):997–1018, 2010.

[12] S. Kolski, et. al. Autonomous driving in structured and unstructured environments. In *Intelligent Vehicles Symposium, IEEE*, 2006.

[13] K. Konolige, et al. Outdoor mapping and navigation using stereo vision. In *Proc. ISER*, 2008.

[14] Y. Kuwata, et al. Real-time motion planning with applications to autonomous urban driving. *Control Systems Technology, IEEE Transactions on*, 17(5):1105–1118, 2009.

[15] J. Leonard, et al. A perception-driven autonomous urban vehicle. *JFR*, 25(10):727–774, 2008.

[16] J. Michels, et. al. High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proc. ICML*, 2005.

[17] M. Montemerlo, et al. Junior: The stanford entry in the urban challenge. *JFR*, 25(9):569–597, 2008.

[18] U. Muller, et. al. Off-road obstacle avoidance through end-to-end learning. In *Proc. NIPS*, 2005.

[19] D. Pomerleau. Defense and civilian applications of the ALVINN robot driving system. In *Gov. Microcircuit Applications Conf.*, 1994.

[20] D. Silver, J.A. Bagnell, and A. Stentz. Applied imitation learning for autonomous navigation in complex natural terrain. In *FSR*, 2009.

[21] D. Stavens, et. al. Online speed adaptation using supervised learning for high-speed, off-road autonomous driving. In *IJCAI*, 2007.

[22] S. Thrun, et al. Stanley: The robot that won the DARPA grand challenge. *JFR*, 23(9):661–692, 2006.

[23] C. Urmson, et al. Tartan racing: A multi-modal approach to the DARPA urban challenge. Technical Report CMU-RI-TR-, 2007.

[24] D. Wettergreen, et. al. Developing nomad for robotic exploration of the atacama desert. *Robotics and Autonomous Systems*, 26(2):127–148, 1999.